Adaptive Multi-Constraints in Hardware-Software Partitioning for Embedded Multiprocessor FPGA Systems

TRONG-YEN LEE[1], YANG-HSIN FAN[1,2] and CHIA-CHUN TSAI[3]
[1]Graduate Institute of Computer and Communication, National Taipei Univ. of Technology, Taipei, Taiwan, ROC
[2]Dept. of Computer Science and Information Eng., National Taitung Univ., Taitung, Taiwan, ROC
[3]Dept. of Computer Science and Information Eng., Nanhua Univ., Chia-Yi, Taiwan, ROC
tylee@ntut.edu.tw, yhfan@nttu.edu.tw, chun@mail.nhu.edu.tw; http://www.ntut.edu.tw/~tylee

Abstract: An embedded multiprocessor *field programmable gate array* (FPGA) system has a powerful and flexible architecture that the interaction between hardware circuits and software applications. Modern electronic products, such as portable devices, consumer electronics and telematics, can be evaluated rapidly in this platform via the implementation of a set of hardware and software tasks. However, the functionality is markedly increased, resulting in a significant raise in the number of hardware and software tasks. Consequently, too large of a solution space is formed to achieve hardware-software partitioning. Moreover, a partitioning result with low power consumption and fast execution time is difficult to obtain since meeting simultaneously multi-constraints from hundreds of thousands of combinations of hardware-software partitions is difficult. Thus, this work presents a hardware-software partitioning scheme that can obtain a partitioning result that satisfies multi-constraints from massive solution space. Specifically, this study attains a partitioning result with low power consumption and fast execution time. The effectiveness of the proposed approach is demonstrated by assessing a JPEG encoding system and a benchmark with 199 tasks.

Key-Words: - Adaptive multi-constraints partitioning, hardware-software partitioning, embedded multiprocessor system, FPGA system, hardware-software codesign

1 Introduction

Many embedded multiprocessor *field programmable gate array* (FPGA) systems, such as the Xilinx [1] ML310 FPGA, have been manufactured since the fabrication process exponentially increases transistor capacity. Many electronics products are developed on an embedded multiprocessor FPGA system, which has the following advantages. First, flexible hardware and software architectures provide various computing abilities that meet performance requirements. Second, the floating operation is more easily implemented by running applications on a processor platform than on a hardware circuit. Third, various functions associated with hardware and software tasks can be evaluated rapidly. Finally, the time and money required for system implementation are reduced compared to those for *application specific integrated circuits* (ASICs). Consequently, embedded multiprocessor FPGA systems have been adopted by both industry and academia. Industrial applications include the design and verification of portable devices, consumer electronics and telematics. In term of academic research, the fields are hardware-software partitioning, hardware-software codesign and hardware-software cosynthesis.

Hardware-software partitioning can be applied to identify a specific functional element that is then implemented as either a hardware or software task. This partitioning is useful when determining the roles of hardware or software for various functional elements while developing embedded multiprocessor FPGA systems. Therefore, many studies [2]-[22] focused on hardware-software partitioning and related issues. First, each functional element must be determined such that it can be implemented by hardware and software tasks. Second, a system can be implemented successfully by combining the identified hardware and software tasks. Third, the completed system must satisfy simultaneously all system constraints such as power consumption, execution time, memory size, slice capacity and concurrency ability. Finally, low power consumption and/or fast execution time can be achieved by the system. This work presents a hardware-software partitioning approach that generates a partitioning result for a complete system and, simultaneously, satisfies multi-constraints. Particularly, the proposed approach achieves low power consumption and fast execution time on embedded multiprocessor FPGA systems.

The remainder of this paper is organized as follows. Section 2 describes the hardware-software partitioning problem for embedded multiprocessor FPGA systems. Preliminary work for hardware-software partitioning is presented in Section 3. Section 4 then introduces the hardware-software partitioning approach to overcome the problems of adaptive multi-constraints. Experimental results for two design examples, a JPEG encoding system and a benchmark case with 199 tasks, are presented in Section 5. Finally, conclusions are given in Section 6.

2 Problem Description

Challenges associated with hardware-software partitioning are that constraints are too varied to coordinate and the solution space is too large to find. Power consumption, execution time, resource allocation and/or concurrency ability are typical constraints when developing embedded multiprocessor FPGA systems. These challenges make attaining a partitioning result that satisfies multi-constraints simultaneously by hardware-software partitioning difficult. Conversely, combining markedly increased hardware and software tasks generates an extremely large solution space that significantly increases the time complexity of hardware-software partitioning.

2.1 Multi-constraints

This work considers five constraints—power consumption, execution time, memory size, slice capacity and number of processors. The power-consumption constraint is the limitation of total power dissipation after hardware-software partitioning. An excessive number of hardware tasks generally increases power dissipation, which may exceed the power-consumption constraint. Thus, an appropriate combination of hardware and software tasks can to comply with the power-consumption constraint. The execution-time constraint limits the time for all task routes. Hardware tasks usually have faster execution times than software tasks. Hence, if a partition result has many hardware tasks, a design has fast execution time; however, power consumption increases. This phenomenon forms a trade off between power consumption and execution time. The constraints of memory size and slice capacity are FPGA resources for implementing hardware and software tasks, respectively. A large memory size can increase the capacity for various computations of software tasks. Conversely, an increased number of hardware tasks require an increased number of slices for implementation. These two constraints comprise another trade-off problem. The constraint of concurrency depends on the number of processors that can perform multiple software tasks simultaneously in a multiprocessor FPGA system. If a design has two processors, such as a MicroBlaze and PowerPC processor, two software tasks can be executed simultaneously. These five constraints, which are called multi-constraints, must satisfy demand simultaneously while performing hardware-software partitioning.

2.2 Solution Space

The number of tasks and constraints are two factors that determine solution space. Current embedded multiprocessor FPGA systems have rapidly increased the number of tasks, as the functions for required by modern electronic products has increased rapidly. If a system has only 1 task, it can be implemented via hardware or software. Therefore, the solution space comprises 2 (*i.e.*, 2^1) partitioning combination for the system. For a design consisting of fifty tasks, the solution space increases to 1125899906842624 (*i.e.*, 2^{50}) partitioning combination, i.e. excessively too large for hardware-software partitioning. This study estimated that the time required to find the optimum result for hardware-software partitioning was over 100 days using an Intel Core Duo CPU with 1GB RAM. *Equation* (1) shows the solution space, which depends on the number of tasks.

$$\text{Solution space} = 2^n,\ n \in 1, 2, \ldots, N \qquad (1)$$

where n is the sum of tasks, N is natural number.

A design for 1 constraint is much simpler than that for multi-constraints because only 1 factor, such as low power consumption, fast execution time, efficient resource allocation or concurrency ability from solution space, is considered. Such a design does not have a trade-off problem. However, modern electronic products are developed under multi-constraints. Consequently, a design has significantly increased complexity as both tasks and constraints proliferate. Once a design must deal with more than two constraints, hardware-software partitioning becomes a complex problem with a trade-off problem between low power consumption and fast execution time, as well as small memory size and high slice usage. The solution space for 1 (labeled X1) to 5 (labeled X5) constraints is huge—from 1E+15 to 6E+15 (Fig. 1).

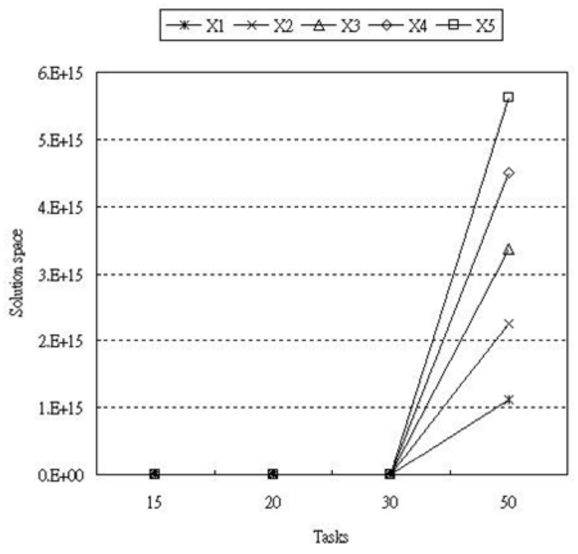

Fig. 1 Solution space of five constraints for 50 tasks

3 Preliminary Work

A task is an atomic unit in an embedded multiprocessor FPGA system. Each task can be implemented as a hardware component or software procedure when developing an embedded multiprocessor FPGA system. In other words, each task has two types of implementation. The hardware-software partitioning technique is a decision system that determines each task to be implemented as hardware or software. Thus, an embedded multiprocessor FPGA system consists of a set of tasks that will be partitioned into two sets of hardware, set *H* and one software set, set *S*, after hardware-software partitioning.

This work developed a partitioning tool shown in Fig. 2-5 which based on the Kernighan and Lin [18] algorithm. The partitioning tool can construct a bi-sectioned and balanced system of sets *H* and *S*. Additionally, the developed tool can reduce external cost via swapping a subset from *H* and with one from *S*. Figure 2 shows a case with 10 randomly generated tasks (*i.e.*, Nos. 0-9) comprised of *H* (left) and *S* (right) sets. This case has two costs: internal and external costs. An arc connects tasks within *H* or *S*; these arcs are called internal costs. External cost defines the cross connection between *H* and *S* tasks. Prior to partitioning, the external cost is 15 (Fig. 2). Figure 3 shows the partitioned result with a reduced cost of 8, which involves the exchange of four tasks (refer to No1 to No6, and No2 to No8 in Fig. 3), respectively. This case indicates that cost is reduced by 46.67%.

This work discusses the un-weighted effect and weighted effect on internal and external costs in the developed tool. Figure 4 shows a case with 24 tasks with weighted costs randomly created on the internal and external cost. Figure 5 shows that the cost is reduced by 23.62%. Although the ability of the developed tool was extended from the un-weighted effect to the weighted effect, a drawback is that only 1 constraint is considered. Thus, such studies may be inapplicable to modern electronics requirements. Therefore, additional constraints, such as power consumption, execution time, resource allocation and concurrency ability, should be further considered in hardware-software partitioning research.

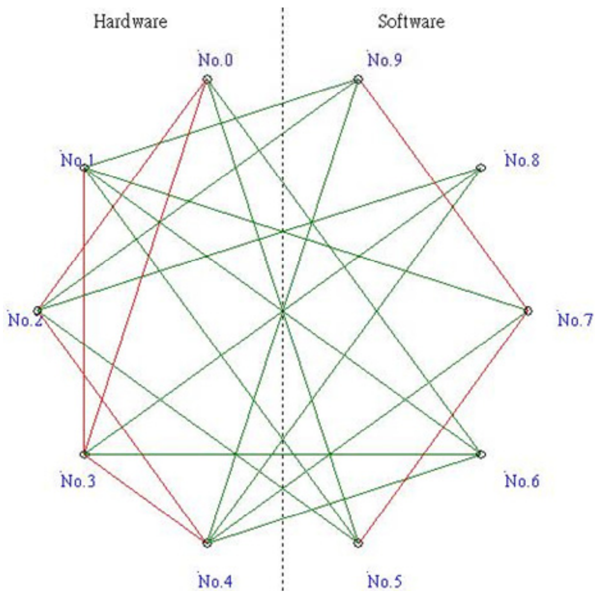

Fig. 2 A bi-section and balanced system for randomly generated 10 tasks

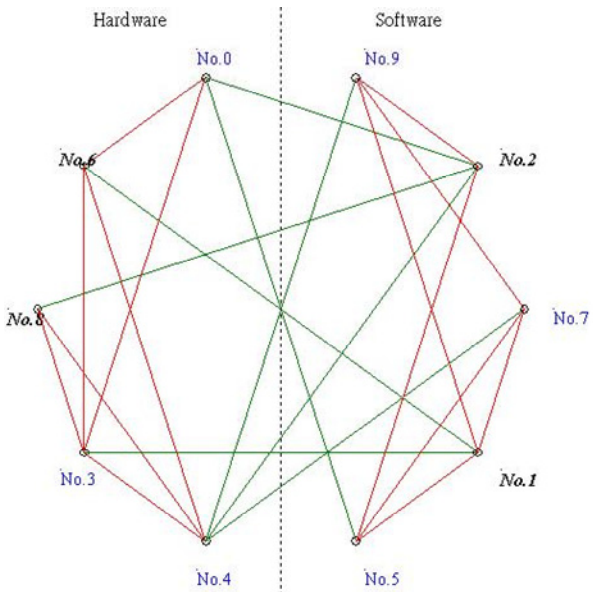

Fig. 3 Reduced external cost after partitioning process for bi-section and balanced system

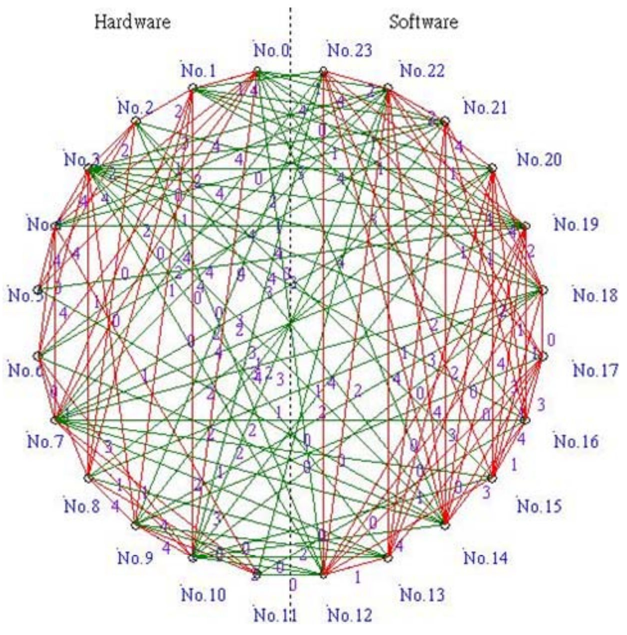

Fig. 4 A weighted bi-section and balanced system for randomly generated 24 tasks

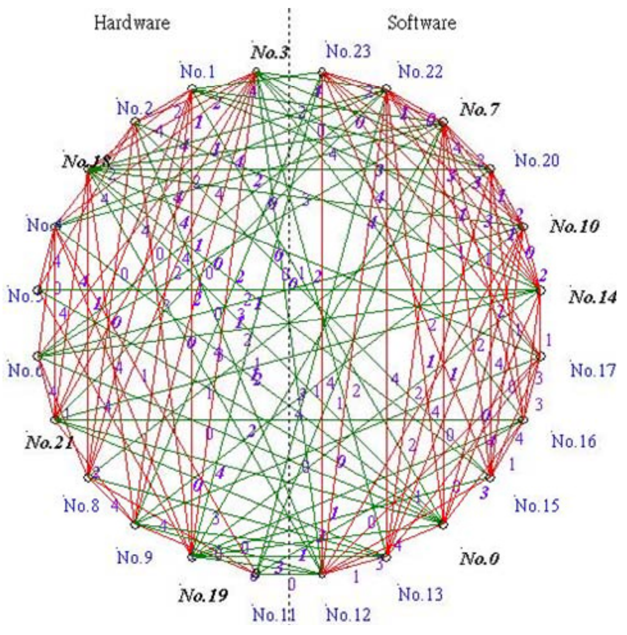

Fig. 5 Reduced external cost for a weighted bi-section and balanced system

Our previous studies [19]-[22] extended one constraint for the cost of hardware-software partitioning to multi-constraints, thus satisfying the market requirements. These approaches include enhancement partitioning [19], hardware-oriented partitioning [20], efficient partitioning [21] and sophistically computing partitioning [22]. However, the achievement of low power consumption and fast execution time does not discuss among these approaches.

4 Partition Approach for Adaptive Multi-constraints

The multi-constraints for the proposed approach are power consumption, execution time, memory size, slice capacity and number of processors. The proposed approach is valuable in practice for meeting simultaneously multi-constraints. Additionally, achievement of low power consumption and fast execution time enhances the value of the proposed approach. However, a fast execution time typically results in increased power consumption in embedded multiprocessor FPGA systems as the number of tasks implemented by hardware increases. In other words, the constraint of power consumption and execution time are difficult to meet simultaneously. Other constraints related to memory size and slices capacity correspond to efficient utilization of FPGA resources. Memory size determines computing capacity when a software task is performed. Reduced memory utilization results in a partitioning result with few software tasks. On the other hand, implementing hardware tasks requires slices to synthesize the circuit. That is, additional slices are required when a partitioning result consists of many hardware tasks. Hardware-software partition must consider the utilization of resources when partitioning software and hardware tasks. The other constraint, concurrency ability, is considered by multiprocessor. These multi-constraints have trade-off relationships with each other, such as that between low power consumption and fast execution time, and small memory size and high slice usage. Notably, a design must have multi-constraints with a trade-off problem that may not involve power consumption and execution time, as well as memory size and slice size. Therefore, this work aims adaptive multi-constraints for the proposed hardware-software partitioning approach that attains a good partitioning result. Additionally, power consumed by the proposed approach is low and execution time fast for embedded multiprocessor FPGA systems.

4.1 Assumptions

In focusing on hardware-software partitioning issues for embedded multiprocessor FPGA systems, we make eight assumptions. First, we assume any embedded multiprocessor FPGA system can be divided into a set of tasks that can be modeled by a task graph. Second, all tasks can be implemented by hardware and software. Third, the constraints of power consumption, execution time, memory size and slice capacity for each hardware task and software task can be measured. Fourth, hardware tasks consume more power consumption than software tasks. Fifth, hardware tasks have faster execution time than software tasks. Sixth, implementing a hardware task only requires slices, as opposed to software tasks the merely needs memory. Seventh, interface cost is zero. Finally, every processor in a multiprocessor has the same characteristics implying that each processor has same computation for each software task.

4.2 Task Graph

Figure 6 displays a task graph used to model embedded multiprocessor FPGA systems while developing various designs. The task graph is a 3-tuple set, $G(V, E, L)$, where V is a set of tasks consisting of hardware and software tasks, E is a set of arcs connecting tasks, and L is a set of levels representing the height of the task graph. The task graph can be used to compute the solution space, describe system behavior and analyze various partitions. The sum of solution space can be derived via *Equation* (1) once the task graph is developed. System behavior is associated with events labeled by arcs. The quality and quantity of any partitioning result can be analyzed using a task graph when multi-constraints are satisfied.

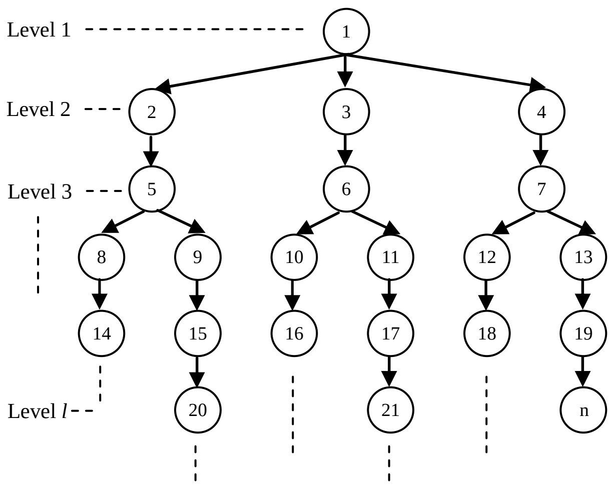

Fig. 6 A task graph of *n* tasks and *l* levels

4.3 Partitioning Method

A huge solution space and too many coordinates for multi-constraints are two challenges in hardware-software partitioning. This work overcomes these challenges and generates a partitioning result that has two benefits. First, all system constraints can be satisfied simultaneously after hardware-software partitioning. Second, low power consumption and fast execution time are achieved by applying the hardware and software tasks to an embedded multiprocessor FPGA system.

The first advantage, that all system constraint can be satisfied simultaneously, is attained by sequentially and individually overcoming constraints. The order of multi-constraints starts with power consumption, followed by execution time, memory size, slice capacity and number of processors. The power consumption constraint is considered first to attain the following benefits. 1) Some tasks can be treated as software tasks as the power consumed by hardware tasks is too high, indicating that low power consumption is achieved because tasks with high power consumption exceed the power-consumption constraint. 2) Software tasks have a high probability of becoming candidates in partitioning results because low power consumption is required. 3) Most of the partitioning results can be filtered by power consumption than other constraints. We can infer that power consumption is based on the largest amount of computation tasks among all constraints. Execution time is considered after power consumption. Once the power-consumption constraint is met, software tasks generally have a high probability to be candidates of partitioning results as they consume little power. Thus, the candidates in partitioning results may consist of more software tasks than hardware tasks. This phenomenon will reduce execution time. Consequently, this work improves execution time by reducing the number of software tasks. This can be achieved by considering the execution-time constraint as the candidates of partitioning results can be filtered. Moreover, partitioning results will not skew toward software tasks, and then coordinate with hardware and software tasks rather than all hardware tasks or all software tasks. This meets the objective of hardware-software partition. Allocating appropriate resources is the next goal. Thus, many software tasks in partitioning results will be filtered out since allocated memory is insufficient. Similarly, the proposed approach can filter out some partitioning results as slice capacity cannot afford to implement too many hardware tasks. Finally, concurrency ability for performing software tasks simultaneously is evaluated using the constraint for deployment of processors.

A set of steps are implemented as follows to obtain a set of partitioning results with satisfying all constraints simultaneously. Power consumption constraint is overcome in the beginning in the following. First, we suppose an embedded multiprocessor FPGA system is modeled using a task graph consisting of *m* processors, *n* tasks and *l* levels. The number of *m* software tasks can be performed simultaneously for each level *l*. Moreover, the number of all hardware tasks is *n*. Similarly, the number of all software tasks is also *n*. Thus, two sets, hardware set *H* and software set *S*, can be formed. Second, hardware set *H* and software set *S* are sorted separately using the power-consumption constraint. Third, a new class, *C*, with 2*n* elements is constructed by merging the sorted hardware set and software set for computing the power consumed by various partitioning results. Fourth, a set of partitioning results, R_p, that meets the power-consumption constraint become a candidate of partitioning results for overcoming other constraints. For example, the sum of power consumed from element 1 to *n* of *C* is calculated to evaluate the partitioning system using all hardware tasks. Next, element 1 of a hardware task is swapped with element 1 of a software task, which indicates that the system is partitioned by hardware task 2 to *n* and software task 1. Thus, the partitioning result consists of element 2 to (*n*+1) of *C*. The swap procedure is repeated until element (*n*+1) to 2*n* of *C* is assessed. Consequently, the number of (*n*+1) partitioning results, which incorporate all hardware tasks, half of the hardware and software tasks, and all software tasks, are evaluated.

Other hardware-software partitioning constraints are overcome sequentially and individually using the following steps. First, a new set of sorting that meets the power-consumption constraint generated from R_p. Second, the procedure for overcoming the execution-time constraint starts with the smallest power consumption and moves to the greatest power consumption. This procedure is repeated until each partition of R_p is evaluated. Then, a set of partitioning results, R_e, is obtained that simultaneously meets the execution-time and power-consumption constraints. Similarly, a set of partitioning results, R_{mem}, that meets the memory-size constraint can be derived from R_e. Additionally, a set of partitioning results, R_s, that meets the slice-capacity constraint can be derived from R_{mem}. Arbitrary processors, such as dual, quad or higher, can be designed. Once the multiprocessor constraint is determined, a set of partitioning results, R_m, that meets multi-constraints is obtained. According to this discussion, Theorem 1 is derived.

Theorem 1: If a feasible partitioning exists in a task graph, then a set of partitioning results that satisfies multi-constraints of power consumption, execution time, memory size, slice capacity and multiprocessor simultaneously will be obtained.

Proof: For a system of task graph with *m* processors, *n* tasks and *l* levels is shown in Fig. 6. The solution space S_s is 2^n. A set of $a_{i,}$ *i*=1, 2, …, *n* represents the hardware tasks sorted according to their power consumption. A set of $a_{j,}$ *j*=(*n*+1), …, (2*n*-1), 2*n* represents the software tasks sorted according to their power consumption. *Equation* (2.1) consists of a_i and a_j, which represent hardware set *H* and software set *S*, respectively. *Equation* (2.2) indicates that the system is implemented using *n* hardware tasks. *Equation* (2.3) indicates that the system is implemented using (*n*-1) hardware tasks and 1 software task. *Equation* (2.4) indicates that the system is implemented using (*n*-2) hardware tasks and 2 software tasks. Thus, the (*j*+1) partitioning result is comprised of *n* software tasks. A set of partitioning results, R_p, that meets power-consumption constraint can be determined using *Equation* (3) where P_{spec} is the power consumption constraint.

$$a_1, a_2, \ldots, a_{n-1}, a_n, a_{n+1}, \ldots, a_{2n-1}, a_{2n} \tag{2.1}$$

$$R_p(1)= a_1+a_2+, \ldots,+a_{n-1}+a_n \tag{2.2}$$

$$R_p(2)= a_2+, \ldots, +a_n+ a_{n+1}= R_p(1)- a_1+ a_{n+1} \tag{2.3}$$

$$R_p(3)= R_p(2)- a_2+ a_{n+2} \tag{2.4}$$

...

$$R_p\,(j+1)= R_p(j)- a_n+a_{2n} \tag{2.5}$$

$$\{\, R_p \mid R_p \leqq P_{spec,}\ R_p \subset S_s \,\} \tag{3}$$

The execution-time constraint is considered after power consumption. A set of partitioning results, R_e, can be determined using *Equations* (4) and (5) from R_p. Symbol $E_{max}(j)$ is the slowest execution time for a task in each level *j*; E_{spec} is the execution-time constraint.

$$R_e=\sum_{j=1}^{l} E_{max}(j) \tag{4}$$

$$\{\, R_e \mid R_e \leqq E_{spec}, R_e \subset R_p \} \tag{5}$$

The memory-size constraint is considered after the execution-time constraint. A set of partitioning results, R_{mem}, can be determined using *Equations* (6) and (7) from R_e, where $C_{sw}(j, i)$ is memory used by task i and level j. Symbol C_{FPGA_sw} is the memory-size constraint.

$$R_{mem} = \sum_{j=1}^{l}\sum_{i=1}^{n} C_{SW}(j,i) \tag{6}$$

$$\{ R_{mem} \mid R_{mem} \leqq C_{FPGA\_sw}, R_{mem} \subset R_e\} \tag{7}$$

The slice-capacity constraint is considered after the memory-size constraint. A set of partitioning results, R_s, can be determined by *Equations* (8) and (9) from R_{mem}, where $C_{hw}(j, i)$ is the slice utilization in task i and level j. Symbol C_{FPGA_hw} is the slice-capacity constraint.

$$R_s = \sum_{j=1}^{l}\sum_{i=1}^{n} C_{HW}(j,i) \tag{8}$$

$$\{ R_s \mid R_s \leqq C_{FPGA\_hw}, R_s \subset R_{mem}\} \tag{9}$$

The number of processors is considered after the slice-capacity constraint. A set of partitioning results, R_m, that satisfies the constraint of number of processors for each level j can be determined by *Equations* (10)–(12), where $R_m(j)$ is a set of partial partitioning results for each level j, and $M_{spec}(j)$ is the number of processors in each level j.

$$R_m(j) = \begin{cases} 1, \text{ if } m = n = 1, \\ 2^n, \text{ if } m = n \text{ and } m > 1, \\ 2^n - \sum\limits_{i=m+1}^{n} C_i^n, \text{ if } m<n \text{ and } m>1 \end{cases} \tag{10}$$

$$R_m(j) \leqq M_{spec}(j) \tag{11}$$

$$\{ R_m \mid R_m \leqq M_{spec}, R_m \subset R_s \} \tag{12}$$

□

A set of partitioning results that satisfies multi-constraints simultaneously is obtained, and a partitioning result with low power consumption and fast execution time can be achieved by Theorem 2.

Theorem 2: If a set of partitioning results is obtained that satisfies multi-constraints of power consumption, execution time, memory size, slice capacity and number of multiprocessors simultaneously, a partitioning result with low power consumption and fast execution time can be attained.

Proof: For a set of partitioning results, R_m, that meets multi-constraints in an embedded multiprocessor FPGA system, a set of partitioning results, O_p, with sequential power consumption can be determined using the sorting technique. A set of partitioning results, O_e, with sequential execution time can also be derived based O_p. Therefore, a partitioning result with low power consumption and fast execution time can be acquired.

□

4.4 Partition Algorithm

Figure 7 presents the proposed algorithm of hardware-software partitioning, HW-SWPartition(), for obtaining a partitioning result with low power consumption and fast execution time. First, the sum of power consumed by various partitioning results is computed. Then, a set of partitioning results, PassPower(), that meets power consumption constraint is determined by comparing the power consumption constraint with their power consumed.

Algorithm HW-SWPartition(*n*)

<u>**Input:**</u> a power consumption of hardware tasks sorted $a_1, a_2, \ldots, a_n$ and software tasks sorted $a_{n+1}, a_{n+2}, \ldots, a_{2n}$

<u>**Output:**</u> Low power consumption and fast execution time partitioning result

```
{
1  i=1;
2  For (j=n to 2n+1)
3  { If (Sum[a_i, … , a_j] ≦ P_spec) then
4     { PassPower(i)=Sum[a_i, … , a_j];
5        i++;
6     }
7  }
8  While (QuickSort(Sum(PassPower(k))) ≦ E_spec)
9     { PassExec(k)=PassPower(k); }
10 While(QuickSort(Sum(PassExec(x))) ≦ C_FPGA_sw)
11    { PassMem(x)=PassExec(x); }
12 While(QuickSort(Sum(PassMem(y))) ≦ C_FPGA_hw)
13    { PassSlice(y)=PassMem(y); }
14 While (QuickSort(Sum(PassSlice(z))) ≦ M_spec)
15    { PassAll(z)=PassSlice(z); }
16 LowPowerFastTime(PassAll);
}
```

Fig. 7 Hardware-software partitioning algorithm for embedded multiprocessor FPGA system

```
Procedure LowPowerFastTime (PassAll);
{
1   //Find average power consumption from PassAll
2   $P_{ave}$=AveragePower(PassAll);
3   //Find average execution time from PassAll
4   $T_{ave}$=AverageTime(PassAll);
5   While (power of PassAll($i$)<$P_{ave}$)
6     {
7       If (time of PassAll($i$)<$T_{ave}$) then
8       {
9         LowPowerFastTime =PassAll($i$);
10      }
11    }
}
```

Fig. 8 Procedure for finding partitioning result of low power consumption and fast execution time

Next, a set of sequential execution time is generated by QuickSort(), and then a set of partitioning results that meets the execution-time constraint is obtained. This procedure is repeated until memory size, slice capacity and number of processor are derived. Finally, a partitioning result with low power consumption and fast execution time is obtained by the function LowPowerFastTime().

The running time of HW-SWPartition() in Fig. 7 is O(n+1) for label 2–7. Label 8–9 is O(k×$m$log$m$) where O($m$log$m$) is derived from QuickSort(). Similarly, Label 10–11, 12–13 and 14–15 is O(x×$m$log$m$), O(y×$m$log$m$) and O(z×$m$log$m$), respectively. Thus, the running time is O(n+k×$m$log$m$) from label 2–15 in Fig. 7. Another function, LowPowerFastTime(), in Fig. 8 is O(l), where l is the number of partitioning results satisfying all constraints. In summary, the time complexity of HW-SWPartition() is O(n+k×$m$log$m$).

5 Experimental Results

This work uses the *joint photographic experts group* (JPEG) [23][24] encoding system and a 199-tasks benchmark [25] to demonstrate the feasibility of the proposed scheme. The experimental platform is a P4, 2.8GHz PC with the memory of 1GB and a Xilinx ML310 FPGA emulation board.

The measured data and task graph of first experiment, which uses the JPEG encoding system, were presented by Lee [19]. The power-consumption constraint is set at 600mW, that is, the total power consumed by hardware in 1 second. Other constraints, slice capacity of 13696 and memory size of 2448, are for two embedded processors.

Table 1 shows a comparison of the proposed method, GA [8], Lin [17], HOP [20] and GHO [21]. The execution time of proposed method shown in column 2 is 20021.66×10^{-6} seconds, faster than that of the GA [8], Lin [17] and HOP [20]. On the other hand, the GHO [21] had the same execution time as the proposed approach. The proposed method shown in column 3 consumes less power than HOP [20] and GHO [21]. Therefore, the proposed approach has the fast execution time and low power consumption.

As a well-defined benchmark is not supplied by academia or industry, Purnaprajna [25] *et al*. presented a case with 199 tasks and a task graph generated randomly to simulate an embedded system. This study implements a graphical user interface that generates a task graph randomly (Fig. 9) and applies the proposed method. Figure 9 presents the embedded system with 199 tasks. The evaluation parameters include execution time and power consumption. The execution time parameters for hardware and software tasks, 0–4 and 0–30, respectively, were generated randomly. The power-consumption parameter for hardware and software tasks was generated randomly, and was 0–3 and 0–1. This experiment was performed five times (column 1 in Table 2). Columns 2 and 3 in Table 2 lists experimental results obtained using the proposed method. Columns 4 and 5 in Table 2 show the average execution time and power consumption computed to determine the effectiveness of the proposed method. Columns 6 and 7 in Table 2 show the case 1 results of obtained by Purnaprajna [25]. Columns 2 and 4 in Table 2 show comparison results for execution time, indicating that the proposed method has the fastest execution time for all cases. In terms of power consumption (columns 3 and 5 in Table 2), experimental results indicate that the proposed method for each case has a power consumption lower than the average.

Table 1. Comparison of the proposed method, GA [8], Lin [17], HOP [20] and GHO [21]

Benchmark [19]	Proposed method		GA [8]		Lin [17]		HOP [20]		GHO [21]	
	Time (uS)	Power (mW)	Time (uS)	Power (mW)	Time (uS)	Power (mW)	Time (uS)	Power (mW)	Time (uS)	Power (mW)
JPEG encoding system	20021.66	525.14	20111.26	499.12	20111.26	494.44	20066.64	599.67	20021.66	586.07

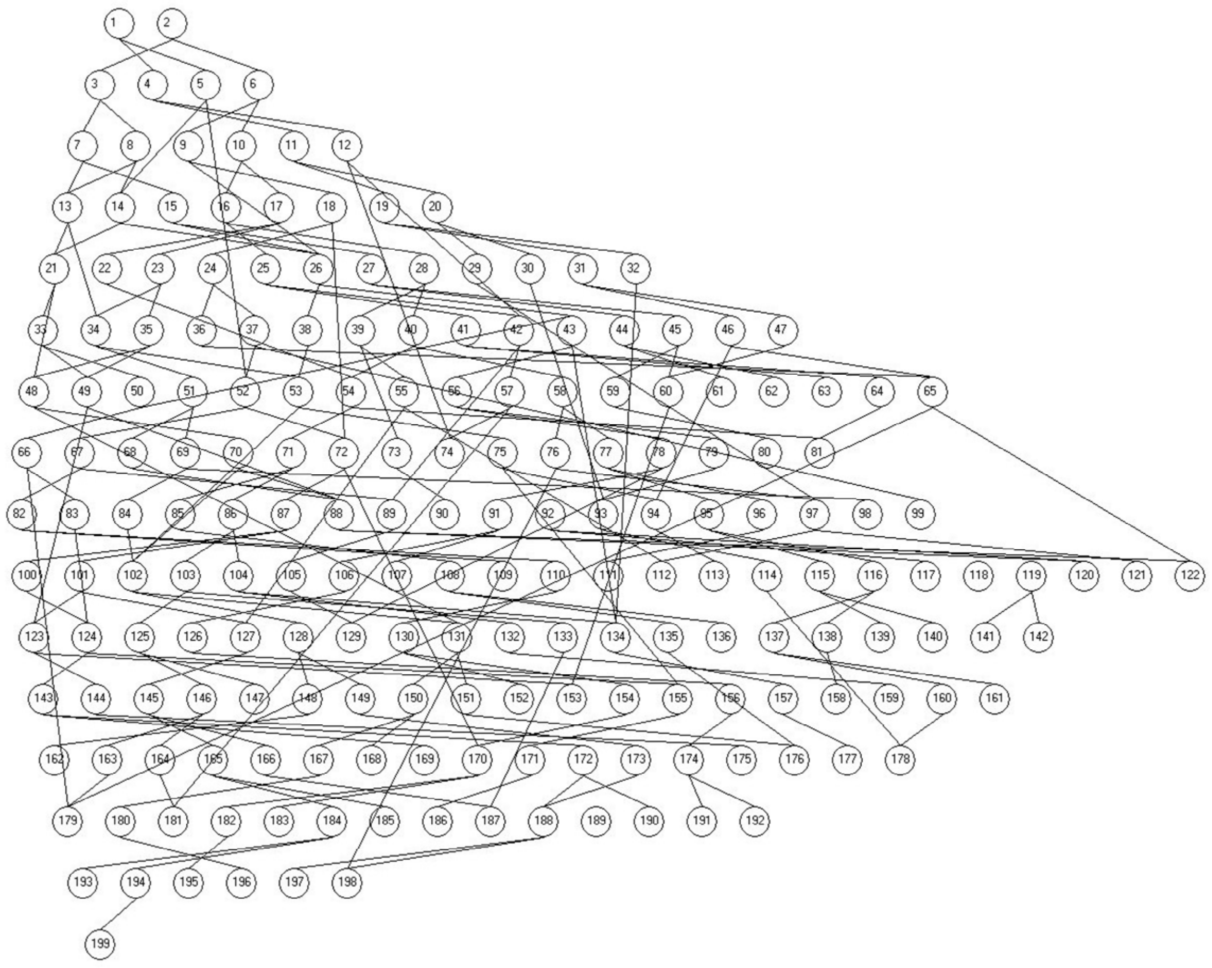
Fig. 9 An embedded system with 199 tasks

Table 2. Comparison of the proposed method and Purnaprajna [25]

Benchmark [25]	Proposed method				Purnaprajna [25]	
	Improved		Average		Case 1: Time and Power	
	Time	Power	Time	Power	Time	Power
Task 199-1	226.17	178.31	230.64	181.35	436.70±1.70	700.50±5.75
Task 199-2	226.17	178.31	230.64	181.35	427.70±4.16	703.80±5.94
Task 199-3	223.80	186.21	231.43	193..42	432.60±5.46	683.00±9.64
Task 199-4	225.86	189.10	226.80	190.66	445.70±8.17	680.10±12.23
Task 199-5	226.17	178.31	230.64	181.35	445.70±7.77	709.40±6.51

6 Conclusions

This work presented a hardware-software partitioning approach that generates a partitioning result the meets multi-constraints of power consumption, execution time, memory size, slice capacity and number of processors simultaneously for an embedded multiprocessor FPGA system. Specifically, the proposed method achieves a partitioning result with low power consumption and fast execution time when the number of hardware and software tasks is increased significantly and constraints are multiple. Two experiments that use a JPEG encoding system and a complex system with 199 tasks demonstrate the feasibility of the proposed method for hardware-software partitioning of embedded multiprocessor FPGA systems.

Acknowledgments The authors would like to thank the National Science Council of the Republic of China, Taiwan, for financially supporting this research under Contract No. NSC-95-2221-E-027-131.

References:

[1] http://www.xilinx.com.

[2] G. F. Marchioro, J. M. Daveau, and A. A. Jerraya, Transformational Partitioning for Co-Design of Multiprocessor Systems, *Proc. of the IEEE Conf. on Computer Aided Design*, 1997, pp. 508-515.

[3] M. Srivastava and R. Brodersen, SIERA: A Unified Framework for Rapid-Prototyping of System-Level Hardware and Software, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 1995, pp. 676-693.

[4] C. J. N. C. Jr, D. C. D. S. Jr and A. O. Fernandes, Hardware-Software Codesign of Embedded Systems, *Proc. of the XI Brazilian Symposium on Integrated Circuit Design*, 1998, pp. 2-8.

[5] R. Ernst, J. Henkel and T. Benner, Hardware-Software Cosynthesis for Microcontrollers, *IEEE Design & Test of Computer*, Vol.10, 1993, pp. 64-75.

[6] V. Srinivasan, S. Govindarajan and R. Vemuri, Fine-Grained and Coarse-Grained Behavioral Partitioning with Effective Utilization of Memory and Design Space Exploration for Multi-FPGA Architectures, *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, Vol.9, 2001, pp. 140-158.

[7] D. Saha, R. S. Mitra and A. Basu, Hardware Software Partitioning Using Genetic Algorithm, *Proc. of the 10th Conf. on VLSI Design*, 1997, pp. 155-160.

[8] Y. Zou, Z. Zhuang and H. Cheng, HW-SW Partitioning based on Genetic Algorithm, *Proc. of the Congress on Evolutionary Computation (CEC2004)*, Vol.1, 2004, pp. 628-633.

[9] H. Kanoh, M. Matsumoto and S. Nishihar, Genetic Algorithms for Constraint Satisfaction Problems, *Proc. of the IEEE Conf. on Man and Cybernetics Systems*, Vol.1, 1995, pp. 626–631.

[10] A. L. Buczak and Henry Wang, Optimization of Fitness Functions with Non-Ordered Parameters by Genetic Algorithms, *Proc. of the Conf. on Evolutionary Computation Congress*, Vol.1, 2001, pp. 199–206.

[11] K. Deb and S. Agrawal, *Understanding Interactions among Genetic Algorithm Parameters: Foundations of Genetic Algortihms 5*, Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[12] S. Palaniappan, S. Zein-Sabatto and A. Sekmen, Dynamic Multi-Objective Optimization OF War Resource Allocation Using Adaptive Genetic Algorithms, *Proc. of the IEEE Conf. on Southeast congress*, 2001, pp. 160–165.

[13] T. Y. Lee, P. A. Hsiung, and S. J. Chen, Hardware-software Multi-Level Partitioning for Distributed Embedded Multiprocessor Systems, *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, 2001, pp. 614-626.

[14] L. Pomante, Co-Design of Multiprocessor Embedded Systems: an Heuristic Multi-Level Partitioning Methodology, *Proc. of the Conf. on Chip Design Automation IFIP*, 2000, pp. 421-425.

[15] C. Brandolese, W. Fornaciari, L. Pomante, F. Salice and D. Sciuto, Affinity-Driven System Design Exploration for Heteregenerous Multiprocessor SoC, *IEEE Trans. on Computer*, Vol.55, No.5, 2006, pp. 508-519.

[16] D. Sciuto, F. Salice, L. Pomante and W. Fornaciari, Metrics for Design Space Exploration of Heterogeneous Multiprocessor Embedded Systems, *Proc. of the Conf. on IEEE/ACM Hardware Software Co-Design*, 2002, pp. 55-60.

[17] T. Y. Lin, Y. T. Hung and R. G. Chang, Efficient Hardware/Software Partitioning Approach for Embedded Multiprocessor Systems, *Proc. of the Conf. on VLSI Design, Automation and Test Symposium* (VLSI-DAT), 2006, pp. 231-234.

[18] Kernighan and Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," The Bell System Technical Journal, Vol. 49, No. 2, 1970.

[19] T. Y. Lee, Y. H. Fan, Y. M Cheng, C. C. Tsai and R. S. Hsiao, Enhancement of Hardware-Software Partition for Embedded Multiprocessor FPGA Systems, *Proc. of the IEEE Conf. on Intelligent Information Hiding and Multimedia Signal Processing*, Vol. 2, 2007, pp. 19-22.

[20] T. Y. Lee, Y. H. Fan, Y. M. Cheng, C. C. Tsai and R. S. Hsiao, Hardware-oriented Partition for Embedded Multiprocessor FPGA Systems, *Proc. of the IEEE Conf. on Second Innovative Computing, Information and Control*, 2007.

[21] T. Y. Lee, Y. H. Fan, Y. M. Cheng, C. C. Tsai and R. S. Hsiao, An Efficiently Hardware-Software Partitioning for Embedded Multiprocessor FPGA System, *Proc. of the International Multiconference of Engineers and Computer Scientists*, pp. 346-351, 2007.

[22] T. Y. Lee, Y. H. Fan, C. C. Tsai and R. S. Hsiao, Sophisticated Computation of Hardware-Software Partition for Embedded Multiprocessor FPGA Systems, *Proc. of the IEEE Conf. on Third Innovative Computing, Information and Control*, 2008.

[23] G. K. Wallace, The JPEG Still Picture Compression Standard, *IEEE Trans. on Consumer Electronics*, Vol.38, 1992, pp. xviii-xxxiv.

[24] K. Kim and J. J. Koh, An Area Efficient DCT Architecture for MPEG-2 Video Encoder, *IEEE Trans. on Consumer Electronics*, Vol.45, 1999, pp. 62-67.

[25] M. Purnaprajna, M. Reformat and W. Pedrycz, Genetic Algorithm for Hardware-Software Partitioning and Optimal Resource Allocation, *J. of Systems Architecture*, Vol.53, No.7, 2007, pp. 339-354.